This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A system for compression, comprising:

a memory device that stores a b-tree data structure comprising a plurality of compressed and uncompressed normalized index keys in sorted order, with no gaps between the stored normalized keys, and stores a plurality of slots with no gaps between the stored slots, wherein the memory device stores the plurality of compressed and uncompressed normalized index keys starting after a header and the plurality of normalized index keys grows towards the end of the memory device as additional index keys are added; and,

a processor that compresses the stored normalized keys, wherein each slot corresponds to a normalized index key in the memory page and comprises a memory offset of the corresponding key and an indicator indicating if the corresponding normalized index key is compressed.

2. (Original) The system of claim 1, wherein the memory device stores the plurality of compressed and uncompressed normalized index keys starting after a header and the plurality of normalized index keys grows towards the end of the memory device as additional index keys are added.

3. (Cancelled)

4. (Cancelled)

5. (Original) The system of claim 1, wherein the processor compresses the stored normalized keys on the memory page by:

(a) determining if a first normalized index key in a memory device should be compressed;

(b) comparing the first normalized index key with a second normalized index key preceding the first normalized index key in the memory device;

(c) generating a common byte length between the first normalized index key and the second normalized index key consisting of the number of bytes in the common prefix between the first normalized index key and the second normalized index key;

(d) replacing the first index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating the memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating the indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed.

6. (Original) The system of claim 5, further comprising the processor repeating steps (a) – (f) for each normalized index key in the memory device.

7. (Original) The system of claim 5, wherein the processor determining if a first normalized index key should be compressed comprises:

examining an indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory device and not compressing a key that does not have a preceding index key on a memory device.

8. (Original) The system of claim 1, wherein the processor compresses the stored normalized index keys before a memory page split.

9. (Currently Amended) A method for compressing a b-tree data structure, ~~compression~~ comprising the following steps:

storing a plurality of compressed and uncompressed normalized index keys of a b-tree data structure in sorted order in a memory page with no gaps between the stored normalized keys;

storing a plurality of slots with no gaps between the stored slots;

storing a header; and,

compressing the stored normalized keys on the memory page, wherein storing the plurality of slots comprises starting immediately at the end of the memory page and growing towards the beginning of the memory page as additional slots are added, further wherein each slot corresponds to a normalized index key in the memory page and comprises of a memory offset of the corresponding key and an indicator indicating if the corresponding normalized index key is compressed.

10. (Original) The method of claim 9, wherein storing the plurality of compressed and uncompressed normalized index keys comprises starting following the header with the plurality of normalized index keys growing towards the end of the memory page as additional index keys are added.

11. (Cancelled)

12. (Cancelled)

13. (Original) The method of claim 9, wherein compressing the stored normalized keys on the memory page comprises:

(a) determining if a first normalized index key in a memory page should be compressed;

(b) comparing the first normalized index key with a second normalized index key preceding the first normalized index key in the memory page;

(c) generating a common byte length between the first normalized index key and the second normalized index key comprising the number of bytes in the common prefix between the first normalized index key and the second normalized index key;

(d) replacing the first index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating the memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating the indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed.

14. (Original) The method of claim 13, further comprising repeating steps (a) – (f) for each normalized index key in the memory page.

15. (Original) The method of claim 13, wherein the determining if a first normalized index key should be compressed comprises:

examining an indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory page and not compressing a key that does not have a preceding index key on a memory page.

16. (Original) The method of claim 9, wherein compressing the stored normalized index keys is performed before a memory page split.

17. (Original) A method for compressing normalized index keys <u>in a b-tree data structure,</u> comprising <u>the following steps</u>:

(a) determining if a first normalized index key in a memory page <u>of a b-tree data structure</u> should be compressed;

(b) comparing the first normalized index key with a second normalized index key preceding the first normalized index key in the memory page;

(c) generating a common byte length between the first normalized index key and the second normalized index key comprising the number of bytes in the common prefix between the first normalized index key and the second normalized index key;

(d) replacing the first index key in the memory page with the generated common byte length followed by the bytes from the first normalized index key that were not in the common prefix between the first normalized index key and the second normalized index key;

(e) shifting the normalized index keys following the first normalized index key to fill any empty memory space resulting from compressing the first normalized index key and updating the memory offsets contained in the slots corresponding to the shifted normalized index keys; and

(f) updating the indicator in the slot corresponding to the first normalized index key to reflect that the key is now compressed.

18. (Original) The method of claim 17, further comprising repeating steps (a) – (f) for each normalized index key in the memory page.

19. (Original) The method of claim 17, wherein the determining if a first normalized index key should be compressed comprises:

examining an indicator in the slot corresponding to the first normalized index key to determine if the first normalized key is already compressed and not compressing a key that has already been compressed; and

determining if the first normalized index key has a preceding index key on the memory page and not compressing a key that does not have a preceding index key on a memory page.

20-23. (Cancelled)